Tracking cloud movement from solar irradiation measurements.

Piotr Parysek

MÁSTER EN INVESTIGACIÓN EN INFORMÁTICA. FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTESNE DE MADRID

Curso académico 2017/2018

Directores:

Daniel Ángel Chaver Martínez
José Ignacio Gómez Pérez

Date of defense: February 22, 2018
Score: 8.5

Autorización de difusión

Piotr Parysek

February 26, 2018

El abajo firmante, matriculado en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "Tracking cloud movement from solar irradiation measurements", realizado durante el curso académico 2017-2018 bajo la dirección de Daniel Ángel Chaver Martínez y José Ignacio Gómez Pérez en el Departamento de Arquitectura de Computadores y Automática, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión y garantizar su preservación y acceso a largo plazo.

Resumen en castellano

En el siglo XXI, la humanidad tiene enormes problemas con la producción de energía. La alta demanda y el uso de energías basadas en residuos fósiles lleva al calentamiento global y la contaminación del aire, la tierra y agua. La energía nuclear también genera mucha controversia tras los accidentes que causaron gran devastación ambiental. Una de las "balas de plata" es la producción de electricidad mediante "fuentes verdes" como viento, aguao o el Sol. Cada uno de ellos tiene sus pros y sus contras. Hoy en día el Sol se está convirtiendo en una de las principales fuentes de energía. Es renovable, respetuoso del medio ambiente y su coste de producción ha disminuido notablemente gracias a la continua mejora de eficiencia de los paneles solares.

Uno de los factores que impiden una mayor difusión de la energía solar es su caracter impredecible. Los actores implicados en la producción y distribución de electricidad necesitan estimaciones precisas de la cantidad de energía que se generará en las próximas horas para satisfacer de forma efectiva la demanda prevista. Asismimo, la gestión de las plantas solares mejoraría con una previsión fiable de la irradación solar en plazos aún más cortos.

Así, en este trabajo, trataré de detecar nubes y determinar su dirección y velocidad. Las nubes filtran gran parte de la radiación solar que llega la superfície, derrumbándose la producción eléctrica, especialmente en paneles fotovoltaicos. Por tanto, automatizar el seguimiento y predicción del movimiento de las nubes es de gran utilidad para la predicción de la energía solar.

Palabras clave

Energía solar; Previsión solar; Irradiancia solar; Detección de movimiento de nube; Medición movimiento de nubes; Pyranometer;

Abstract

In the XXI century humankind has massive problems with producing energy due to the global warming, air, earth, water pollution and "little bit" standing aside from nuclear energy after accidents which caused great environmental devastation. One of the "silver bullets" is producing electricity from "green sources" like wind, water, the Sun. Each of them has its pros and cons. Nowadays the Sun is becoming one of the main power sources. It is renewable, eco-friendly and what is more, does not require many expenses to maintain full functionality. Additionally every year the efficiency of solar panels increases.

One of the main stoppers for solar energy to become widespread is its inherent uncertainty. Companies producing and distributing electricity require accurate estimates of the energy to be produced in the incoming hours in order to efficiently meet user demands. Even shorter forecast horizons are required to operate solar plants.

Thus, in this thesis, I will try to detect clouds and determine its direction and speed. Clouds filter most of solar radiation, heavily impacting solar energy production. Therefore, tracking and predicting clouds movements is of great value for solar energy forecasting.

Keywords

Solar energy; Solar forecasting; Solar radiation; Cloud motion detection; Cloud motion measurement; Pyranometer

Contents

List of Figures

List of Tables

Agradecimientos

Deseo expresar mi agradecimiento al todas mis directores de esta tesis Máster profesor Daniel Ángel Chaver Martínez y profesor José Ignacio Gómez Pérez, por la apoyo que habéis brindado a este trabajo.

Agradezco de mis Profesores desde Polonia por la paciencia y comprensión.

Gracias a mi familia, a mis padres.

Y muchas gracias a todas personas desde Erasmus programe por la gran y inolvidable experiencia y por darme esta oportunidad.

Dedicatoria

Dedico esta tesis a mis padres, todas mis profesores y a todos que me apoyaron.

Chapter 1

Introduction

Nowadays one of the "humankind addiction" is energy. People in well-developed countries generate, use and "waste" a vast amount of electric power. The massive usage of the electricity is visible all around us and people would need to use it in the future. Everyday people have direct or indirect contact with electric devices; for example: when they wake up they turn off the alarm clock in their cell phones, then they make a coffee in their coffee machine, after that they go to work by electric bus or metro (also mostly powered with electricity), they work using computers or any other device, and so on.

In that moment a question arises: where do all these amounts of energy come from? One typical way is to create it in huge power plants, most of which are based on burning fossil fuels like oil, natural gas or coal. The way how it creates energy is simple: the burning fuel heats up water and creates steam. The steam is used to rotate turbines which are connected to generators and that energy is converted into electric energy. Another way is in nuclear plants. After the World War II and military usage of atom bombs in nukes of Hiroshima and Nagasaki in 1945[10], there was a different approach in using this devastating amount of untamed energy. Originally to test the capability and possibilities of this new technology there were created first 'testing' nuclear reactors[3]. These events started 'the time of atom'. 'Supreme' countries, for example: USA, USSR, France, Japan, Korea, built many nuclear power plants due to its capabilities and power in producing energy and to meet the needs and demands of people in their energy 'hunger'.

Unfortunately, each of these methods has its drawbacks. Oil-based methods, even if 'cheap', are assumed to be one of the reasons for The Global Warming[17] which is one of the biggest threats to mankind today, due to its unknown and terrible outcomes. Besides, the price of its production is fluctuating because its fuel is based on the price of oil.

Electricity from nuclear power plants is very expensive to start. Firstly the country which "wants" to build one has to have or buy the necessary technology to safely contain that amount of power. The second drawback is the threat of its malfunction; if an accident happens, it can cause a great natural and environmental devastation. One of the 'atom' history black moments is the Chernobyl accident, which took place on 25 April 1986, where human error and violation of procedures during routine shut down for maintenance caused a steam explosion which led to core meltdown. Another more recent disaster took place at Fukushima, in 11 March 2011 (one of the photos from that accident is shown in the figure: 1.1). This accident happened due to a major earthquake and ulterior tsunami which caused the destruction of power lines and backup generators. Without the external power, the cool down system did not work, leading to a heat decay and reactor damage, including core meltdown and explosive loss of reactor containment[12].

Figure 1.1: Photo which was taken two days after explosion.

To meet the high demands of people and at the same time "save the Earth", it would be better to use "renewable" sources of energy. One example is hydropower plants (see 1.2), where the gravity of falling water runs the turbines to create electricity. But this kind of generating station can be created only near rivers. That kind of producing "green energy" is relatively expensive and it also has big environmental consequences due to damming of water, changing water flow, etc.

Another way of producing electrical energy is by using solar cells. With them, humankind

can create 'clean' energy without emitting carbon dioxide and other 'greenhouse gases' during its production. However, this form of energy depends on our closest star - the Sun. This kind of generating energy is, in its principle, simple: by allowing solar light to flow on silicon-based solar panels, one of the light particles - photon - strikes out a free electron in silicon molecule. The silicon cell thus behaves like a current source, whose intensity largely depends on the incoming radiation. This method sounds simple but in practice it becomes more difficult and tricky, and only 30-40% of that solar light is transformed into energy[11]. Moreover, the energy production is co-related to daytime and weather. At night all of the visible light, photons, are from very distant stars or are deflected from the Sun by the Moon (we do not take into consideration the light created by fire/lamps etc.) and their energy is not sufficient to 'strike out' the electrons and produce enough power. And still, during day, clouds are a big obstacle which deflects or absorbs most of the solar radiation.

Therefore, cloud movement prediction becomes of paramount importance to solar energy industry. In this thesis, I will focus on detection of clouds and prediction of their movement. With that kind of information, solar plants could align their solar panels to maximize energy production and, even more important, they could predict future 'drops' in solar energy production and use other alternatives in order to meet users demands.

Figure 1.2: Photo of Solina Dam one of the biggest in Poland.

1.1 Main goal

The primary goal of this thesis is to prepare and program an autonomous system tor cloud detection and measurement its speed and direction. All the system should work with usage of Internet message broker, for example with MQTT protocol, and should have the ability

to be "remotely controlled". The first draft of the project is presented in figure 1.3 (on page 4).

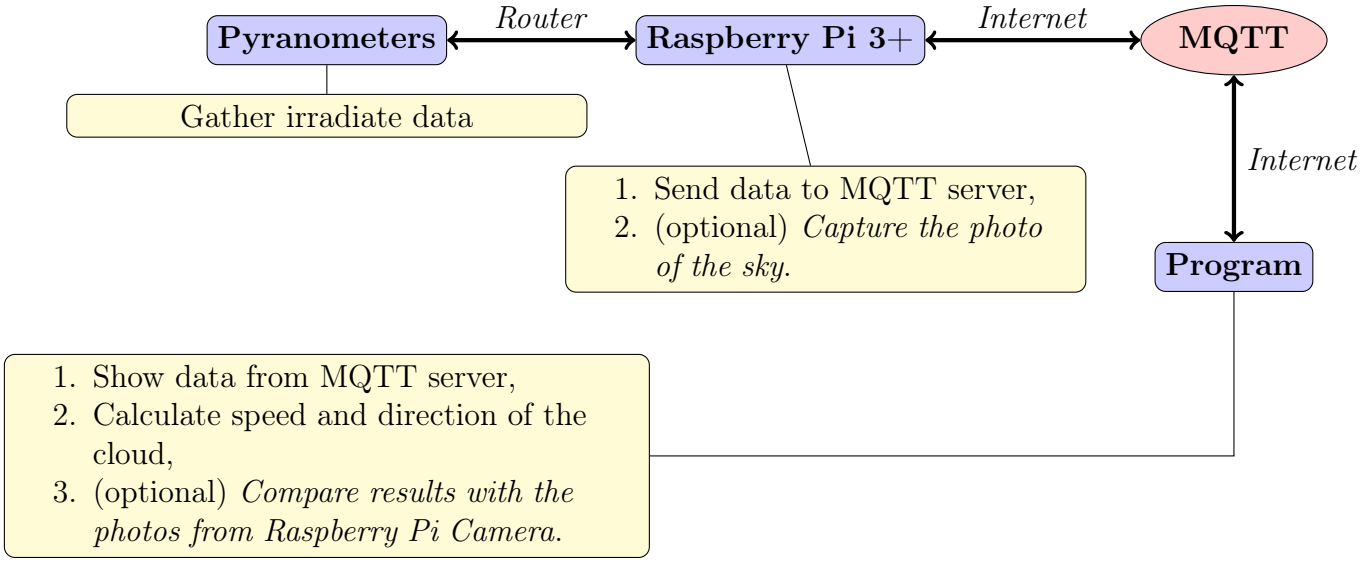


Figure 1.3: Prime project of the system.

Regrettably due to some problems with connection pyranometers with Raspberry the final expression of project changed.

1.2 Contents

There are many methods to detect the presence and movement of the clouds, but each of them requires different tools, equipment, and preparation of its usage. In this thesis, I will present several methods and approaches described in peer-reviewed articles or presented on websites and forums created by and for enthusiasts.

1.2.1 Methods based on images

The most traditional approach for detecting clouds is by making photos of the sky and "expose" clouds. By comparing the photos (taken in a specific time range) it is possible to detect the clouds and extract their direction (and also some kind of density information). Most of the existing work approximates the problem by comparing different values of

color like RGB (Red Green Blue), HSV (Hue, Saturation, Value) and YCbCr (Luminance, Chrominance). Even though this was not the main goal of this theis, we also tried out simple image-based methods in order to compare with the other alternative explored. To execute this task I was given Raspberry Pi 3 with Raspberry Camera Module to capture and filter images.

1.2.2 Methods based on pyranometers

The other method explored to detect clouds is to measure solar irradiation at several close points, so that we can compare the output signal of each node. This method is simple and there have been made many projects based on Arduino[1] to detect a source of light and "follow" it. Mostly these projects are based on photoresistors and "control" the movement of a cell (like a project posted on instructables[18]).

However, photoresistors are not accurate enough to perform the whole detection and tracking. To precisely execute this task I got professional pyranometers, which allowed me to "precisely" detect clouds and their movement.

1.3 Work plan and experimental environment

The project was executed in several phases:

- Documentation on image based cloud tracking proposals,
- Documentation, installation and configuration of OpenCV to implement chosen techniques,
- Documentation on radiation based cloud tracking approaches,
- Experimental environment setup using available pyranometers and data loggers,
- Network setup to remotely access sensed data (images and pyranometers reads),
- Develop Qt based graphical environment to represent and analyze the obtained data.

The final experimental environment consisted on the following equipment:

- Four SKS-1110 silicon based pyranometers,
- Data logger to sample the pyranometers,
- Raspberry Pi 3 with the Raspberry Camera module,
- Wifi Router to switch from data logger local network and *eduroam* to export measurements.

The complete set up, battery operated, has been laid out in the west terrace roof of Physics Faculty building.

Chapter 2

Image based method

The first method which we have implemented in this work was detecting and predicting movement by using images of the sky. This approach seems simple: take pictures, filter "blue color" to get clouds and, by comparing consecutive images, detect the direction of the clouds.

2.1 Method

Most previous works following this approach are based on color segmentation. They differ on the way segmentation is performed; for example, different color spaces are proposed to better remove the sky: RGB, (Red Green Blue) HSV (Hue, Saturation, Value), YCbCr, YIQ.

2.1.1 Color models

In most described methods the key role is to convert each pixel of the image form one color model to another and check the differences between values to determine the desired result - i.e. detect sky or cloud[15],[8].

To convert from standard RGB model to others we can use the following equations:

RGB→ HSV If RGB values are from $0-255$ they are normalized to the range $0-1$

$$R' = \frac{R}{255} \qquad G' = \frac{G}{255} \qquad B' = \frac{B}{255} \tag{2.1}$$

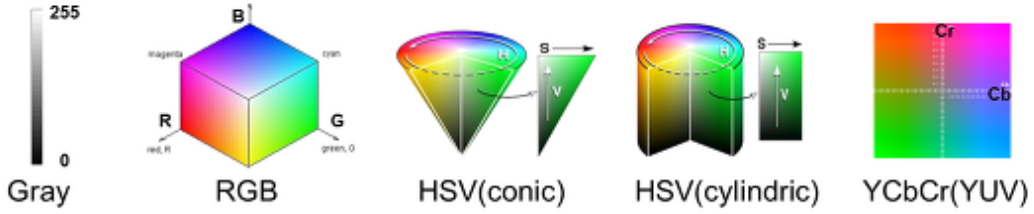


Figure 2.1: Comparison of different color models.

Then auxiliary variables are calculated C_{MAX} and C_{MIN}

$$C_{\max} = \max(R', G', B') \qquad C_{\min} = \min(R', G', B') \qquad \Delta = C_{\max} - C_{\min} \tag{2.2}$$

Then it is possible to calculate different HSV values:

$$H = \begin{cases} O & \textbf{if } \Delta = 0 \\ 60\deg \times (\frac{G'-B'}{\Delta} \mod 6) & \textbf{if } C_{\max} = R' \\ 60\deg \times (\frac{B'-R'}{\Delta} + 2) & \textbf{if } C_{\max} = G' \\ 60\deg \times (\frac{R'-G'}{\Delta} + 4) & \textbf{if } C_{\max} = B' \end{cases} \tag{2.3}$$

$$S = \begin{cases} 0 & \textbf{if } C_{\max} = 0 \\ \frac{\Delta}{C_{\max}} & \textbf{if } C_{\max} \neq 0 \end{cases} \tag{2.4}$$

$$V = C_{\max} \tag{2.5}$$

RGB→ YCbCr This conversion can be written as a matrix multiplication (assuming that RGB values are in the range $0 - 255$):

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{2.6}$$

RGB→ YIQ Analogously to RGB→YCbCr conversion, YIQ can be obtained through simple matrix multiplication:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.7}$$

2.1.2 Camera

Most of the previous works rely on high resolution cameras and / or devices with "fish eye" lenses, which allow us to make images with a wider angle - field of view[2].

However, since this was just a way to check the main method, we used a simple Raspberry Pi Camera Module"[5] original version 1 (shown in figure 2.2). Unfortunately the device has only a 5 Megapixels resolution and, to create clear pictures, we was forced to use VGA resolution (640 × 480).

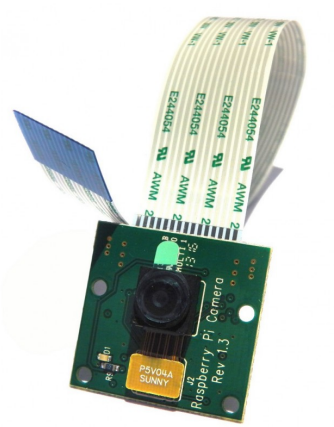


Figure 2.2: Raspberry Pi Camera Board v1.3

2.1.3 OpenCV

In this part of my project we used Open Source Computer Vision Library **OpenCV**[16] version 3.3.1. This library is open source, released under a BSD license, and has huge possibilities and capabilities mainly in real-time computer vision. It is used in many relevant projects in topics such as:

- Recognition systems of facial, gesture, objects;
- Human - computer interactions;
- Robotics;

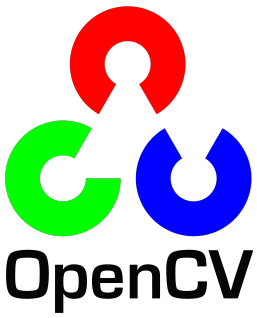


Figure 2.3: OpenCV logo

For example I have used this powerful library in my Engineering thesis: "A system for 3D reconstruction and medical analysis of the lower limbs"[13], where I used it to filter "unnecessary pixels" which were not used to create a 3D model.

2.1.4 Sky filter

To filter the sky in the image and be able to extract the clouds we used several methods found in previous work.

First approach

This method was proposed in a post of the website "OpenCV questions" [14]. The design uses methods already implemented in the OpenCV library.

Firstly it is created a gray scale mask based on the difference between blue and red channels. Then, after normalization Blue - Red Ratio (nbrr) and saturation (sat), the following equations are used:

$$\mathsf{nbrr} = \frac{B - R}{B + R} \tag{2.8}$$

$$\mathsf{sat} = 1 - \frac{\min B, G, R}{\max B, G, R} \tag{2.9}$$

where R, B, G are vales for Red, Green and Blue for each pixel of the image. Subsequently each value is compared as shown in 2.10.

$$\textbf{if}(\mathsf{nbrr} < 0.4 \;\&\&\; \mathsf{sat} < 0.3) \tag{2.10}$$

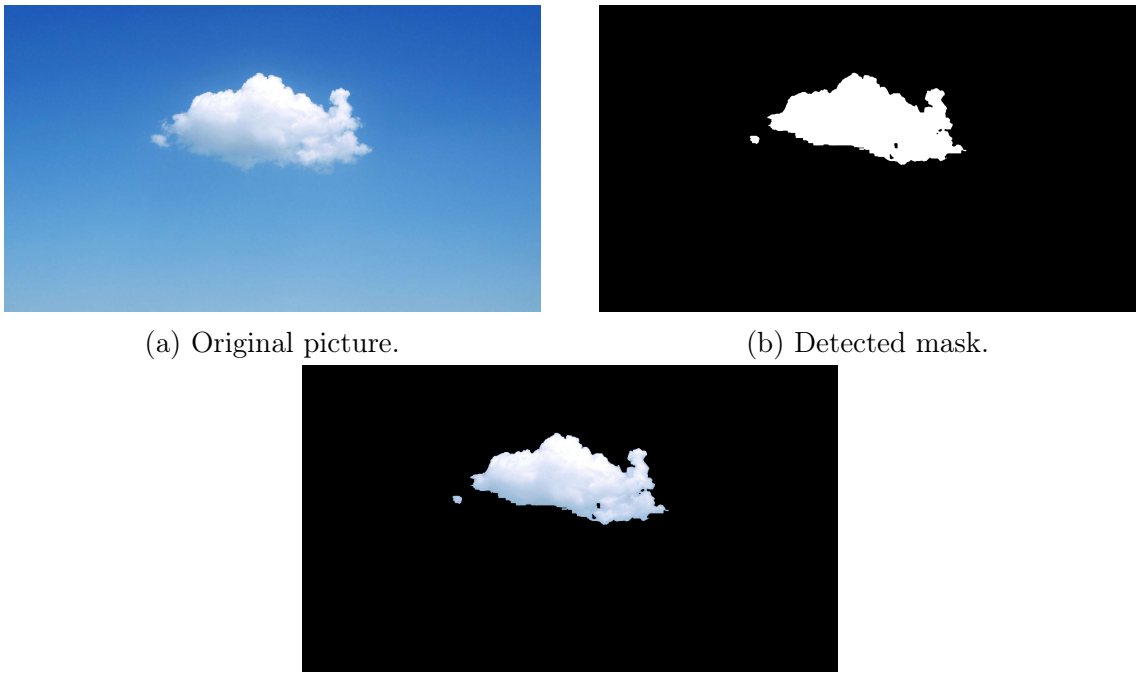
If that condition is fulfilled, the pixel is labeled as cloud; otherwise, it is labeled as sky. Figure 2.4 shows the result of our implementation of this method using an image taken from https://tsicloud.com/2903-2/)

Second approach

Another similar approach is presented in an article published by researchers from universities of Pakistan[6]. This design is very similar to the previous one, but it uses quite different comparisons, as shown in 2.11.

$$\textbf{if}(|R - G| < 5 \;\&\&\; |G - B| < 5 \;\&\&\; B > R \;\&\&\; B > G \;\&\&\; B > 50 \;\&\&\; B < 230) \tag{2.11}$$

For testing this method we have used the same sample picture and the results obtained are shown in figure: 2.5.



(a) Original picture. (b) Detected mask.

(c) Final picture.

Figure 2.4: An example of presented metod.

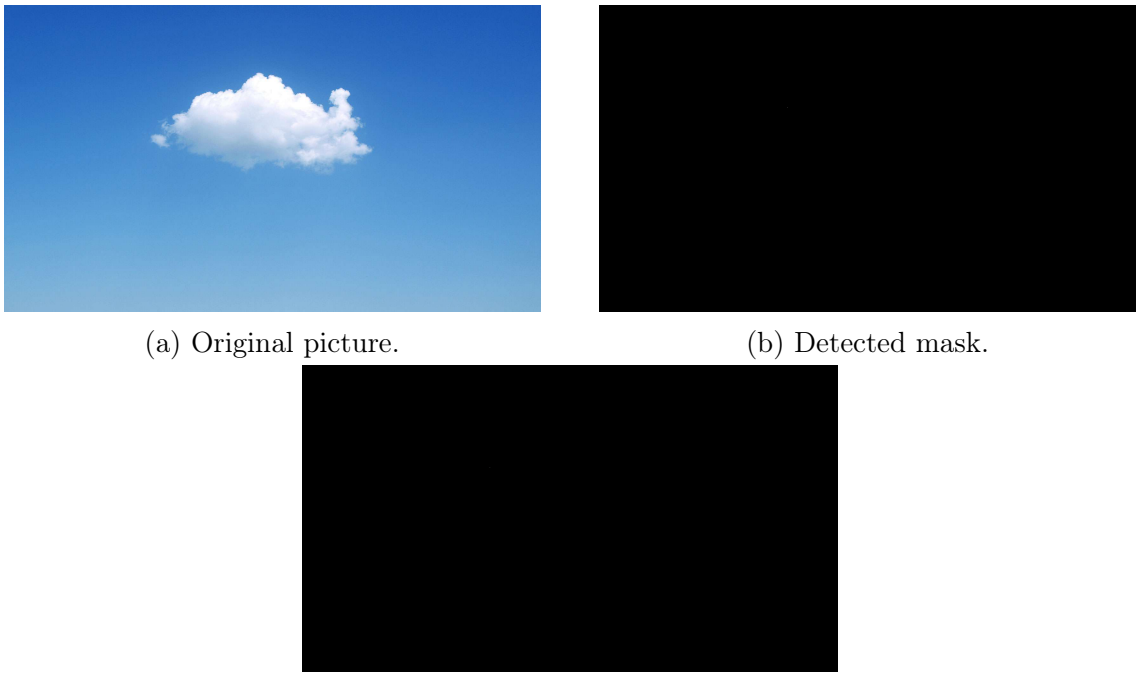
Comparison of the results

As it can be observed in the pictures, the second method behave clearly worse and it caught only one pixel as a 'cloud pixel' (because of the small picture that one pixel is not visible). For this reason, in our experiments we only used the first method.

2.2 Implementation

For this project I could use one Raspberry Pi 3 B Model B V 1.2 (presented in the figure: 2.6) with a simple camera mentioned in subsection 2.1.2. In my opinion this is a very good single-board microcomputer. It has enough computing power (Quad Core 1.2GHz Broadcom BCM2837) and connectivity (BCM43438 wireless LAN and Bluetooth) for our purposes, it has 4 USB 2.0 ports, HDMI and it is equipped with 40 GPIO pins.

I was required to set up an environment that allowed to place the Raspberry Pi on the roof of the Faculty building, collect data (photos and some other data that we will explain



(a) Original picture. (b) Detected mask.

(c) Final picture.

Figure 2.5: An example of presented metod.

later) and send it to a server running in the department and/or cloud storage.

2.2.1 Set up device

To start executing tasks we had to "clean up" the device - install fresh operating system - Raspbian (downloaded and installed according to instructions on: https://www.raspberrypi.org/downloads/raspbian/). After installing the OS, we installed and prepared another steps to properly use the "computer" with instructions described in the next subsections.

Network

The first problem which we had to solve was connecting the Raspberry Pi to the university network - "eduroam" to automatically upload taken pictures to my Google Drive. To do this I had to download installation script for Linux users from the following webpage: https://cat.eduroam.org/. After executing file (by using sudo sh eduroam−linux∗.sh) all

Figure 2.6: Raspberry Pi 3 B Model B V 1.2

the important files were created in directory ~/.cat_installer. In that location I had to copy the content of file cat_installer.conf to the bottom of file /etc/wpa_supplicant/wpa_supplicant.conf. Later I only had to save the files and reboot the board.

Take pictures and send them to Google Drive

Firstly I enabled and connected the Google API with my University Google Account proceeding with instructions and tips published on website: http://jeremyblythe.blogspot.com/2015/06/motion-google-drive-uploader-for-oauth.html. Finally I developed the function shown in listing 2.1 to take and upload pictures.

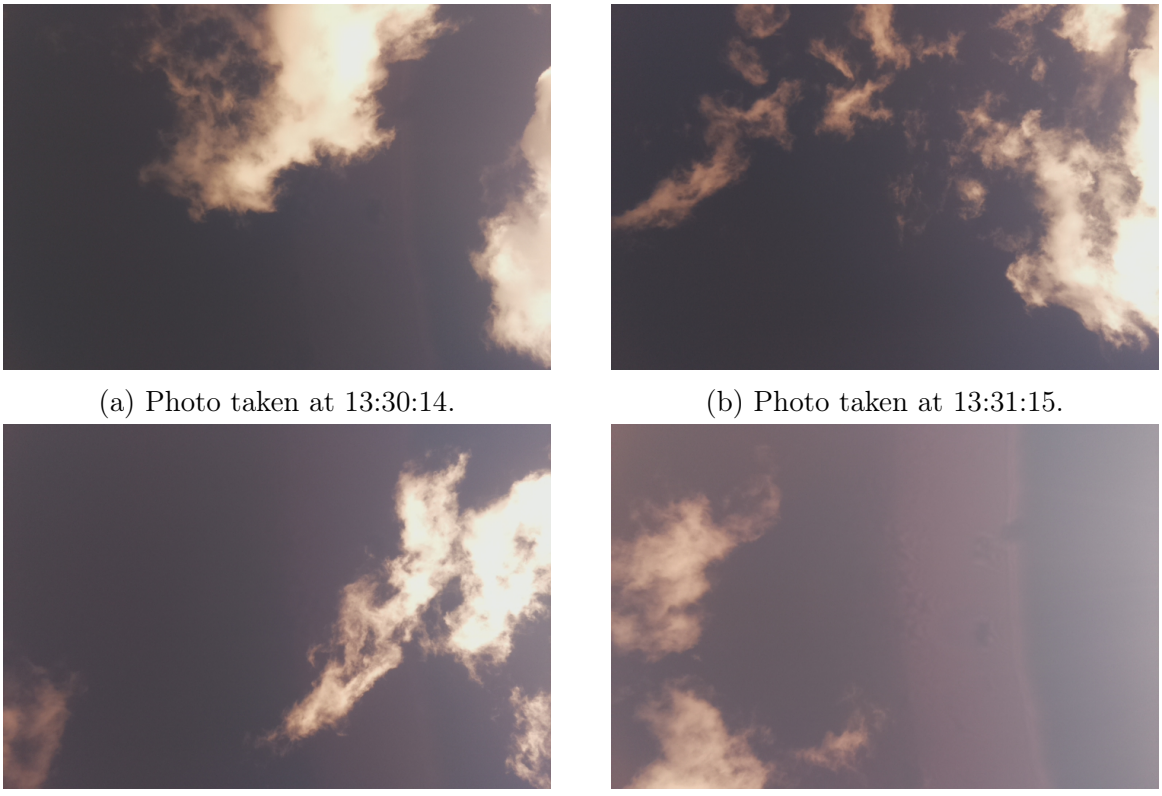
```
def TakePicture():
    camera = PiCamera()
    sleep(8)
    filename1 = time.strftime("%Y%m%d-%H%M%S")
    filename1+=".jpg"
    camera.capture(filename1)
    return filename1
```

Listing 2.1: Function to take picture using Raspberry Pi Camera and returning its name based on current date and time

2.2.2 Results

We have fully tested this part of my project on January 11th and 12th (those days we also measured values with the pyrometers as described in chapter 3 to compare results). In the first day we took one picture every minute. This was not enough granularity, so on the second day we corrected our error and took a picture every 30 seconds.

Figure 2.7 shows raw images of the sky taken on January 11th of 2018. Figure 2.8 shows the images from 13:30 and 13:31 after filtering with the two approaches explained above (Section 2.1.4).



(a) Photo taken at 13:30:14.

(b) Photo taken at 13:31:15.

(c) Photo taken at 13:32:13.

(d) Photo taken at 13:33:14.

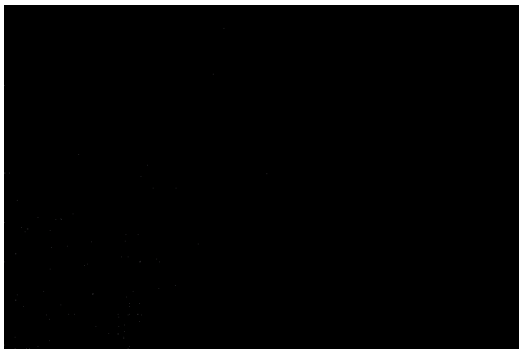
Figure 2.7: Example photos taken in 11-January-2018 at 13:31-34.

Unfortunately based on these photos (original images presented in figure 2.7 and "filtered" photos in figure 2.8) we could not determine the direction of clouds. Marked with black color are pixels of the image marked as a sky, in both methods the aim is to detect

(a) First method applied to image of 13:30:14

(b) First method applied to image of 13:31:15

(c) Second method applied to image of 13:30:14



(d) Second method applied to image of 13:31:15

Figure 2.8: Photos taken in 11-January-2018 at 13:31-34 after "detecting clouds".

sky color and with inverse mask indicate which elements are the desired ones. Basing in the example for "working" photo (figure 2.4 on page 11) in the first method, part of the clouds were detected as sky and "original" sky was omitted. The second method has detected whole pictures as the sky. The described algorithms are not good enough for "unprepared" photos. Moreover, the frequency we set for capturing (1Hz) reveals to be too low to track one cloud on different pictures and efficiently detect its movement.

On the second day (January 12th) we set the Raspberry to take a picture every 30 seconds; example results are shown in the figure 2.9. Unfortunately, due to the weather conditions (there were 100% cloud or 100% clear sky), the algorithms did not work properly and did not obtain any displayable results.

In our opinion, the main problem with obtained pictures was their angle relative to the Sun. Because of that many pixels of images have "too much" "Red particle" in their RGB

color model. The first approach uses normalized Blue - Red ratio and saturation which with these pictures (when the Red amount is bigger than in reality) is have values which do not apply to the equation. Also, this is a cause of wrongly "masking" the sky in the second approach. In that equation (equation 2.11 on page 10) high amount of "Red part" blocks whole statement - that is why whole pictures (with some pixels being exceptions) are marked "as cloud".

2.2.3 Future upgrades

One of the problems with this part of the project was the cheap, low resolution and wrongly aligned camera. Next time we will set up the device in a position to hide sun rays (which are presented for example in the figure 2.7d). Another problem is the field of view. Raspberry Pi camera Module does not have a wide angle of view, so it is difficult to detect the same cloud on different photos at different times.

In my work, we have wrongly chosen the methods to detect clouds. On the clear, "perfect" examples there were no problems and almost 100% detection ratio is obtained (with the "First approach"). However, when it comes to the "real life" images, the defects of methods were patent and in my opinion not acceptable. Also in the future work with this part of the project I can use other methods described in different articles and methods.

Without proper pictures we could not detect movement of objects, but if we had ones we would have used one of the OpenCV methods for example: cv::calcOpticalFlowPyrLK. This function calculates an optical flow based on the pattern of apparent motion. It uses the iterative Lucas-Kanade method with pyramids. Of course this approach makes several assump-

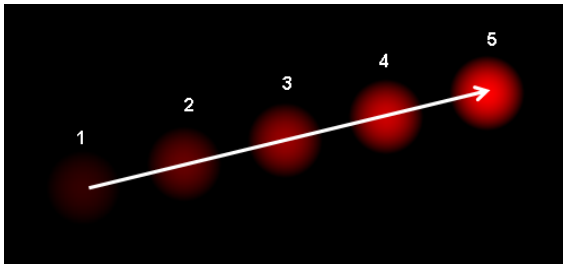


Figure 2.10: Example of optical flow with vector of moving object - desired result of this task.

tions like:

1. The pixel intensities of an object do not change between consecutive frames. That could be a problem, because clouds change their density and shape with time.

2. Neighboring pixels are assumed to have a similar motion.

After solving these problems / difficulties this part of the project could be a great source of information about direction and speed of cloud movement.

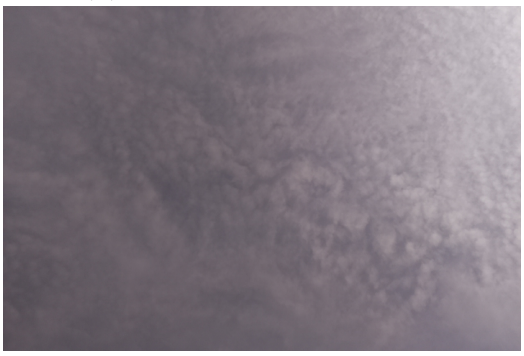
(a) Photo taken at 14:08:12.

(b) Photo taken at 14:08:42.

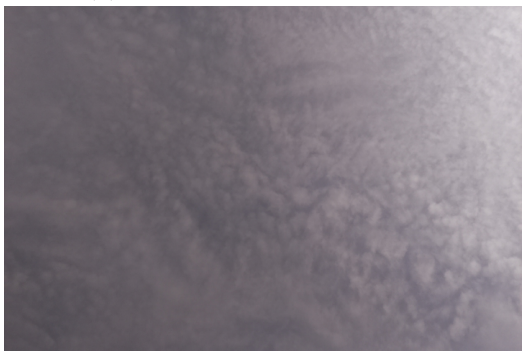
(c) Photo taken at 14:09:11.

(d) Photo taken at 14:09:41.

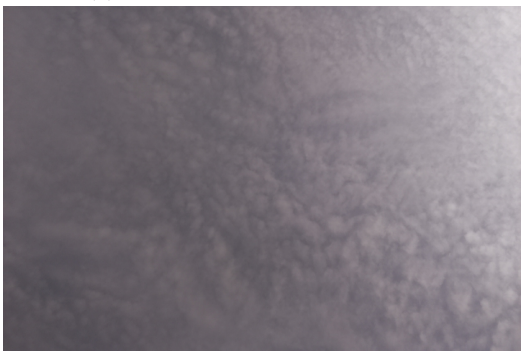
(e) Photo taken at 14:10:12.



(f) Photo taken at 14:10:42.



(g) Photo taken at 14:11:12.



(h) Photo taken at 14:11:42.

Figure 2.9: Example photos taken in 12-January-2018 at 14:08:12 to 14:11:42.

Chapter 3

Pyranometers

Another interesting alternative to detect clouds is to measure the solar irradiation at different (close enough) spots and look for the peaks on the generated signal, as it was presented in Introduction (chapter: 1). However, this approach is not free of drawbacks. It can be imprecise, due to imperfect measurements. Some proposals use simple solar cells to estimate irradiation. The actual measurement is done by timing the charge/discharge of a small capacitor driven by the solar cell. Otherwise, this approach may become too expensive, as the precision depends on the price of the device employed to measure solar irradiation.

To complete this task we used professional and precise Pyranometers sensors: SKS 1110[9]. These devices can detect light wavelength from 350 nm to 1100 nm (typical human eye can detect wavelengths from 390 nm to 700 nm) and has acceptable response time on 10 ns (comparing to devices based on capacitors whose response time is usually higher than 30 ns).

3.1 Methods

We have based our implementation on the work of[7]. The authors use 8 pyranometers placed in a semicircle as shown in Figure 3.1. In the paper, the authors used a radius of $r = 6$m.

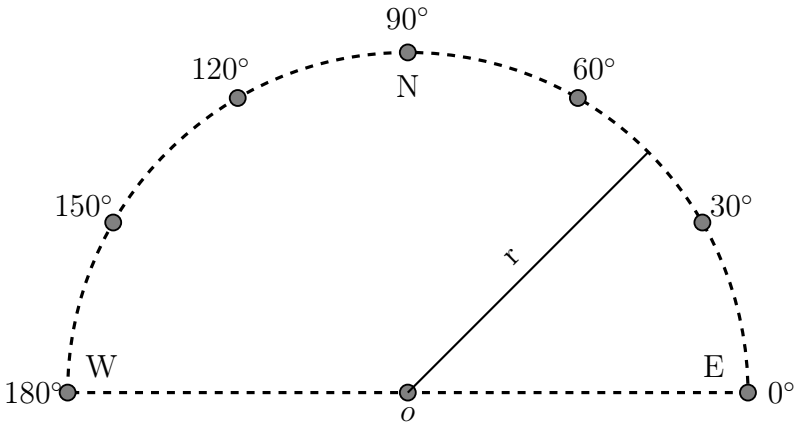


Figure 3.1: Pyranometer setup described in the article, with measured angles about the origin point o and radius $r = 6\mathsf{m}$.

3.1.1 Most Correlated Pair method (MCP)

This is a very basic method, which uses a pair of sensors, for example: S_a and S_b separated by a distance D. The signals output by both sensors should be correlated (given that they are calibrated and not too far from each other) but values may be shifted by a lag t_{ab}, as it is presented in Figure 3.2. Cloud speed can be calculated with equation 3.1.

$$v = \frac{D}{t_{ab}} \tag{3.1}$$

The method to determine the cloud direction correlates the sensors in pairs, with all the pairs including the central sensor (on figure 3.1 marked as o). The pair with the largest correlation coefficient $\rho(T_{ab})$ is assumed to be the most aligned one with the so called *Cloud Motion Vector*, which includes the information of the direction of the cloud.

In the presented configuration the cloud direction error is up to $\frac{30^\circ}{2} = 15^\circ$ and the speed errors up to $1 - \cos 15^\circ = 3.4\%$. Unfortunately, to get these small errors (less than 5%), at least eight sensors are required.

3.1.2 Linear Cloud Edge method (LCE)

Unlike the previous method (on page 20), the second approach presented in[7] uses only a triplet of sensors. For this example we will take sensors labeled as o, E and N from figure

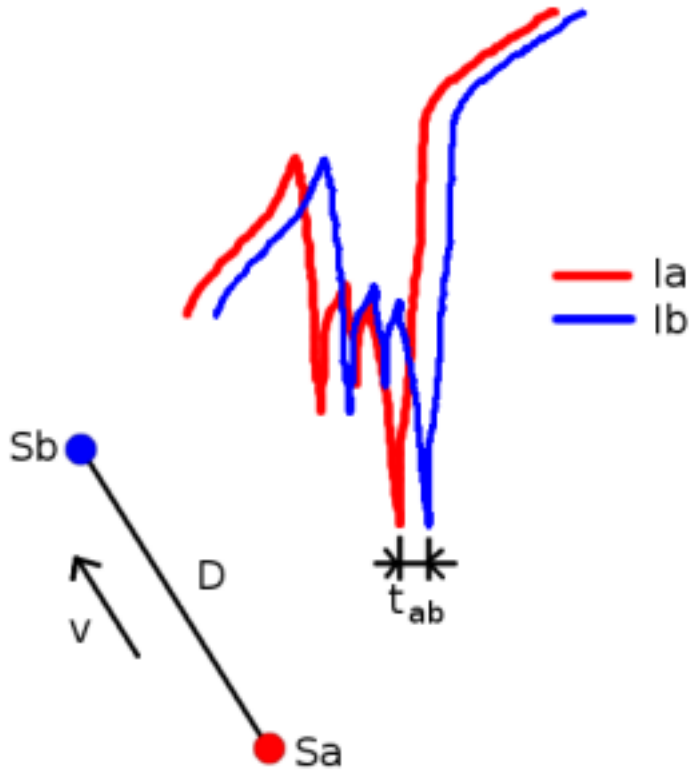


Figure 3.2: Example illustration of output from two sensors S_a and S_b (S_a is the "central one"). Also presented distance D between sensors and change of time in their peaks t_{ab} - all the values needed to calculate the speed of the cloud.

3.1, with the distances as in the article: $\overline{oN} = \overline{oE} = D = 6$,. Precise scheme of this layout is presented in figure 3.3. This method is based on several assumptions:

i) a linear cloud edge shape across the sensor array,

ii) there is a constant Cloud Motion Vector while passing over the three sensors,

iii) the cloud is large enough to cover all three sensors.

To calculate Cloud Motion Vector with this method we need to measure passing time of

at least two clouds, and use them with these equations:

$$\beta = \arctan\left[-\frac{t_{oE}}{t_{oN}}\right] \tag{3.2}$$

$$t_{oN}\sin\alpha + t_{oE}\cos\alpha = \frac{D}{v} \tag{3.3}$$

$$\alpha = \arctan\left[-\frac{t_{oE\ II} - t_{oE\ I}}{t_{oN\ II} - t_{oN\ II}}\right] \tag{3.4}$$

By calculating the value of α from equation 3.4 we can calculate speed of the cloud by using its value in equation 3.3. The value of β, calculated in equation 3.2 is helpful to determine the direction of the cloud.

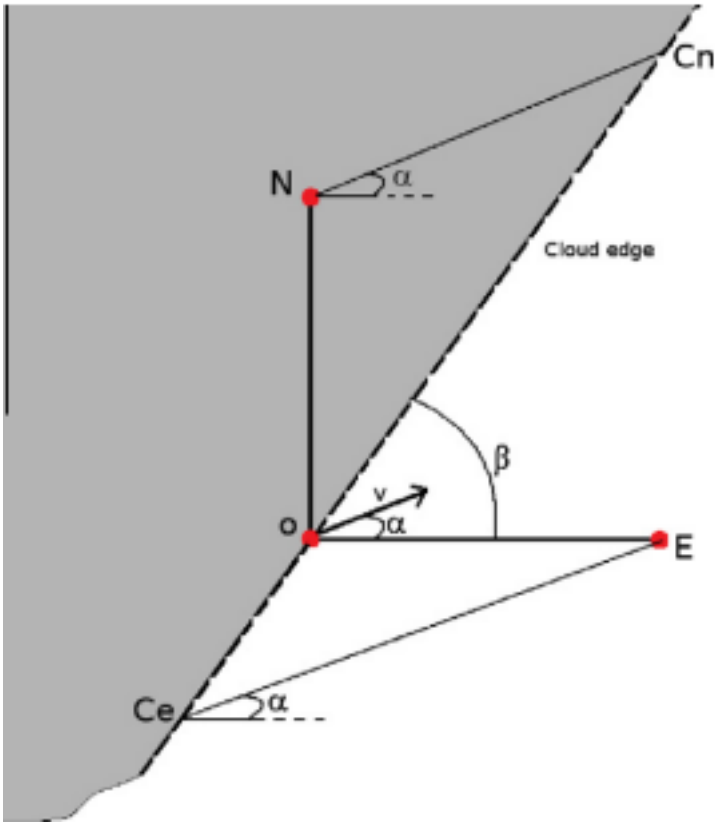


Figure 3.3: Schematic of Linear Cloud Edge, where β is an angle between cloud edge and x axis, α is an angle between Cloud Motion Vector and x axis and C_E, C_N are points in edge cloud that pass over points E and N.

3.1.3 Cloud Motion Components Method (CMC)

This method comes from another article[20] but it is also based on the triple set of sensors, as described in the method from the previous subsection: Linear Cloud Edge (on page 20). In this approach it is also assumed that cloud is big enough to "cover" all the sensors and has constant speed and direction during the time taken to pass. To calculate the speed and angle of the cloud it is mandatory to measure only one cloud passing between oE and oN. To count desired values - Cloud Motion Vector and speed is needed to calculate these equations:

$$V_X = \frac{D}{t_{oE}}; \quad V_Y = \frac{D}{oN} \tag{3.5}$$

$$V_{\text{cloud}} = \sqrt{V_X^2 + V_Y^2} \tag{3.6}$$

$$\alpha_{\text{cloud}} = \arctan\left[\frac{V_X}{V_Y}\right] \tag{3.7}$$

where V_{cloud} is the speed of the cloud and α_{cloud} is direction of the cloud motion.

3.2 Implementation

Given that there were only 4 calibrated pyranometers, we chose the second method - Linear Cloud Edge (described on page 20) for this work.

Figure 3.4: Qt logo.

We have created our project in Qt[?] with C++ language. We used some of the Qt essentials modules (like Qt Core, Qt GUI, Qt Widgets) and also QtCharts to present data.

The method needs to find local maximum and minimum in several time windows, so we used Persistence1D class (downloaded from https://people.mpi-inf.mpg.de/~weinkauf/notes/persistence1d.html[19].

3.2.1 Program

The program has a basic style of executing. Each of the tasks has to be run in direct or indirect order to complete a work - detect clouds, calculate and show their "movement".

1. After starting the program the "main window" will appear as shown in figure 3.5.

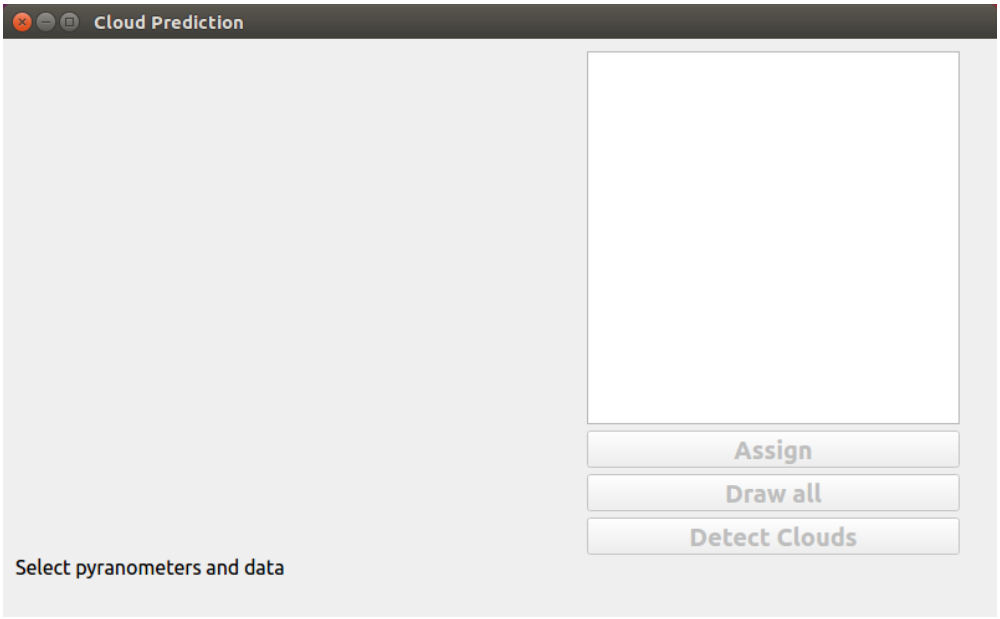


Figure 3.5: Main "clear" window of the program.

2. Then the user has "two options" - the user may firstly set number and location of the pyranometers or select columns data from a file.

   - The user may open a new window to select number and positions of pyranometers by "clicking" in a drop-down menu: File − Set Pyranometers or using keyboard shortcut Ctrl+P. Then in a bordered field the user can add a new pyranometer on the selected location. Pressing a button on existing "pyranometer" will delete it. On figure 3.6 (page 25) it is shown a picture after selection of four pyranometers.

   - To select data from file user has to select from drop down menu: File − Load data from file or use keyboard shortcut Ctrl+F. An example view on "parsed file"

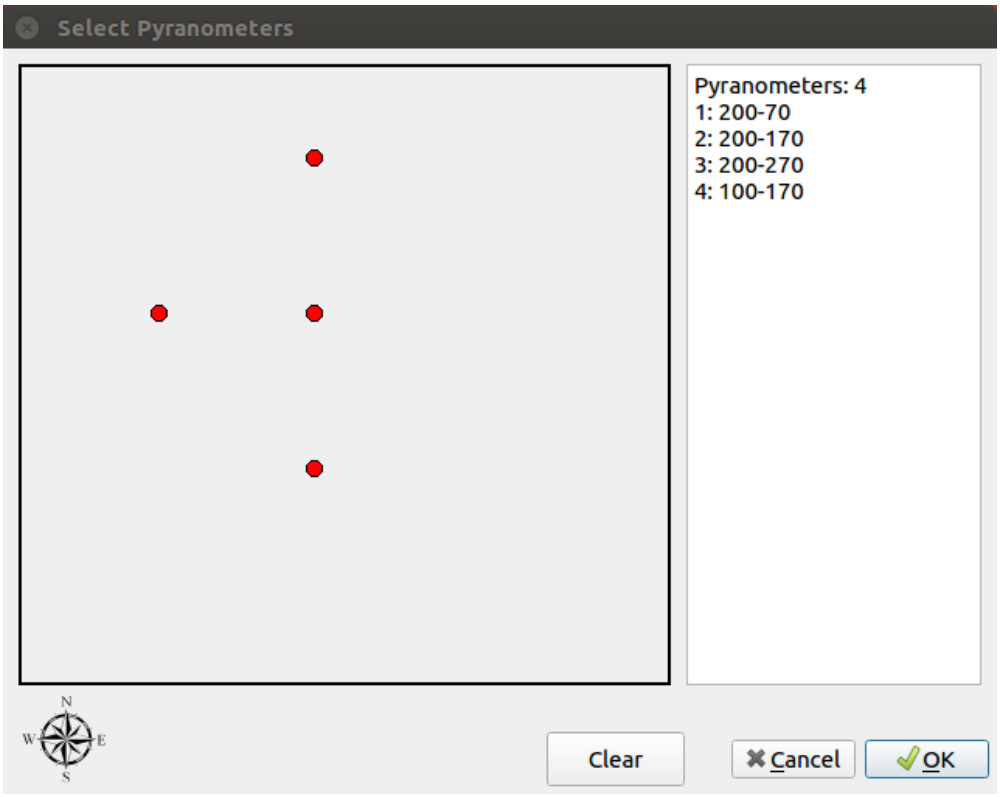


Figure 3.6: Selection of localization of pyranometers.

is shown in figure 3.7 (page 26). User can select whole columns or single cells - it does not matter all data from chosen columns will be taken for later usage, like it is shown in figure 3.8 (page 27).

3. After selection of pyranometers and necessary data (as is shown in figure: 3.9, page 28) the user has to align "relative" position of the pyranometer and certain column with data.

   Name of "columns" are taken from headers of accepted file, but position must be chosen from values: North (**N**), North-East (**NE**), East (**E**), South-East (**SE**), South (**S**), South-West (**SW**), West (**W**), North-West (**NW**), Central (**C**).

   After correct align user can approve adjustment by pressing Asign button in main window, like in figure: 3.10 (on page 28)

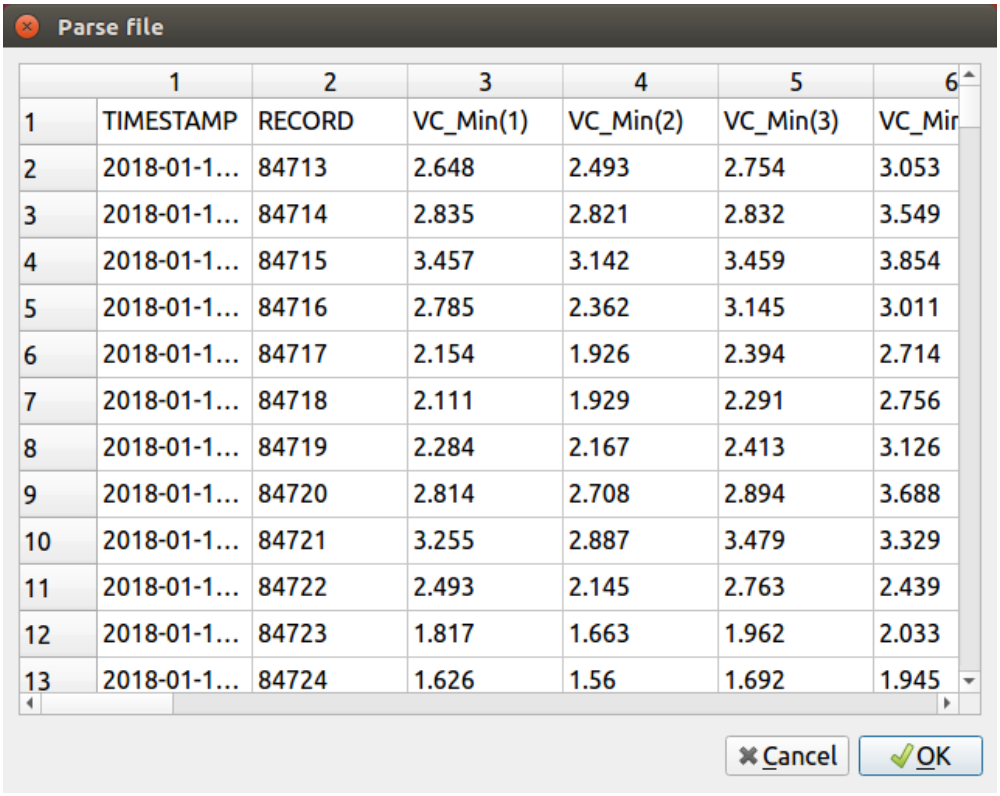


Figure 3.7: View on selected, already parsed file.

4. The successful alignment (presented in figure 3.11 on page 29) will block options to change position of pyranometers or change column adjustment. But then the user can display charts of results for each pyranometer like:

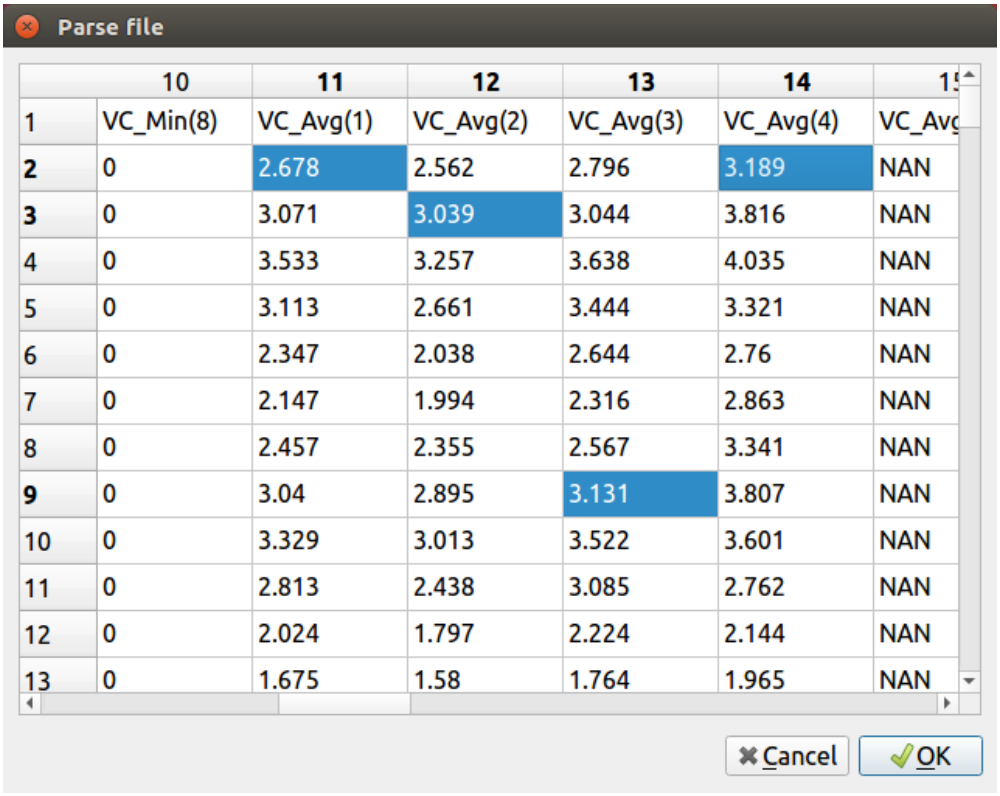
   Pyranometer 1 - presented in figure 3.12 on page 29,

   Pyranometer 2 - presented in figure 3.13 on page 30,

   Pyranometer 3 - presented in figure 3.14 on page 31,

   Pyranometer 4 - presented in figure 3.15 on page 32,

   Every chart displays values based on date and time of measurements. Every chart has the ability to "zoom into" a certain point by left-clicking the desired point or selecting range also with the left mouse button. To move away user has to press right mouse button. Example of this action is presented in figure 3.16 (on page 33).



	10	11	12	13	14	15
1	VC_Min(8)	VC_Avg(1)	VC_Avg(2)	VC_Avg(3)	VC_Avg(4)	VC_Avg
2	0	2.678	2.562	2.796	3.189	NAN
3	0	3.071	3.039	3.044	3.816	NAN
4	0	3.533	3.257	3.638	4.035	NAN
5	0	3.113	2.661	3.444	3.321	NAN
6	0	2.347	2.038	2.644	2.76	NAN
7	0	2.147	1.994	2.316	2.863	NAN
8	0	2.457	2.355	2.567	3.341	NAN
9	0	3.04	2.895	3.131	3.807	NAN
10	0	3.329	3.013	3.522	3.601	NAN
11	0	2.813	2.438	3.085	2.762	NAN
12	0	2.024	1.797	2.224	2.144	NAN
13	0	1.675	1.58	1.764	1.965	NAN

Figure 3.8: View on selection four columns: 11 "VC_Avg(1)", 12 "VC_Avg(2)", 13 "VC_Avg(3)" and 14 "VC_Avg(4)".

5. The user has a possibility to see all gathered data in one chart by pushing button Draw all. The result is presented in figure 3.17 on page 34.

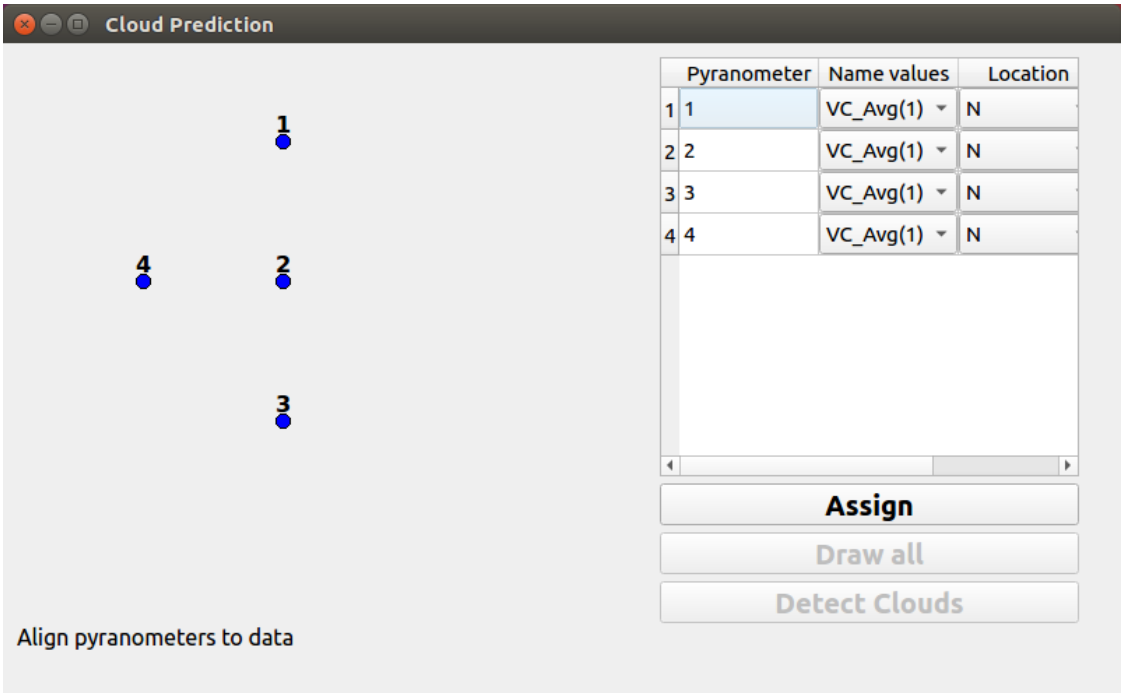


Figure 3.9: "Clear" window after selecting data columns and choosing pyranometers.

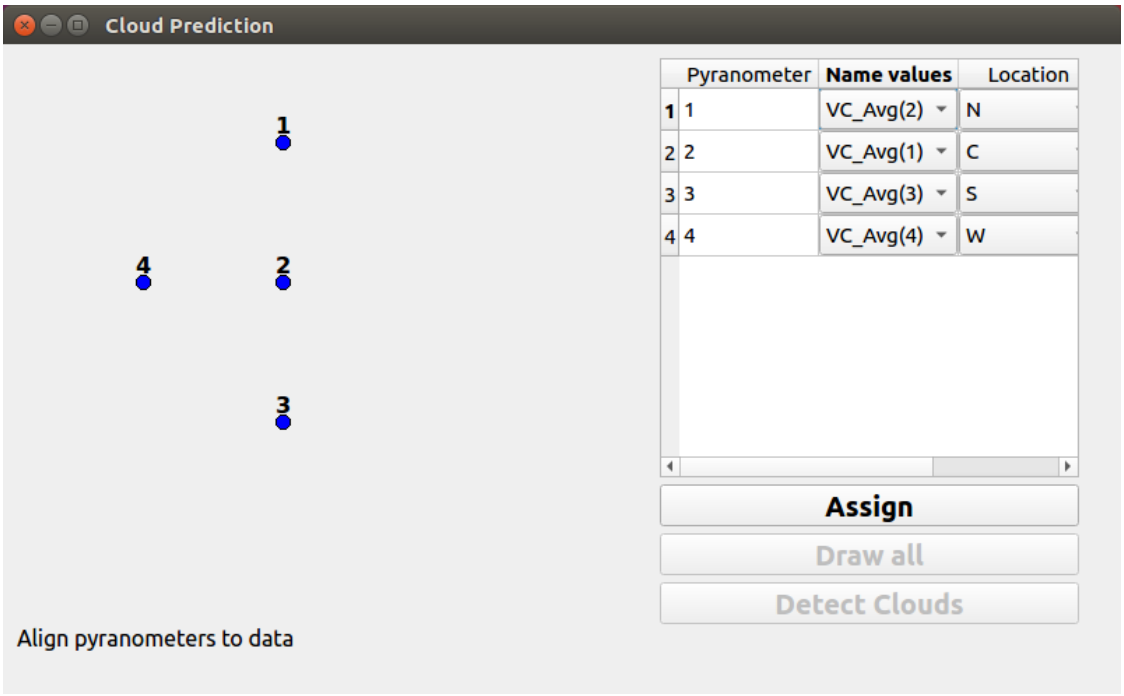


Figure 3.10: Aligned pyranometers with location and data columns - values are ready to be assigned to each other.

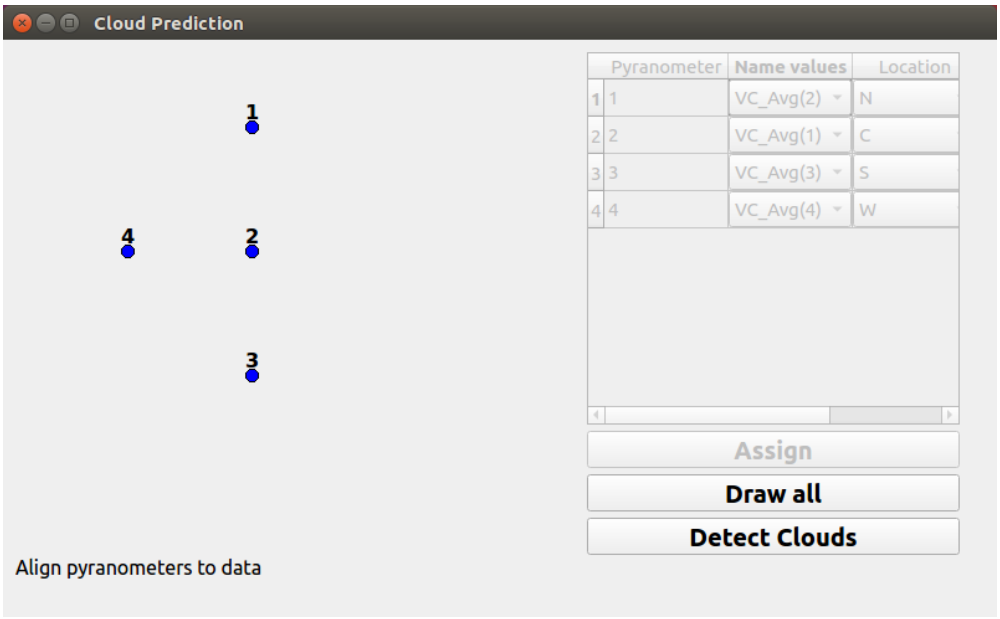


Figure 3.11: Assigned - and blocked for future changes - values. Now user can display charts for pyranometers or "display the cloud movement".

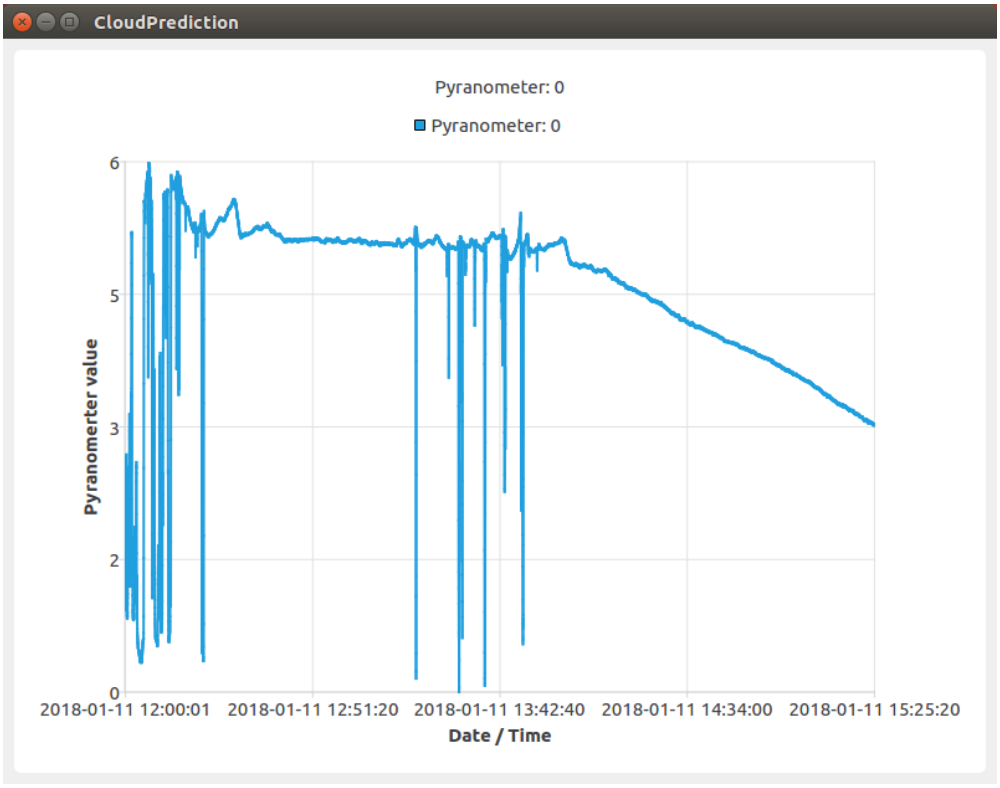


Figure 3.12: Presented data from pyranometer number 1 from 11-January-2018.

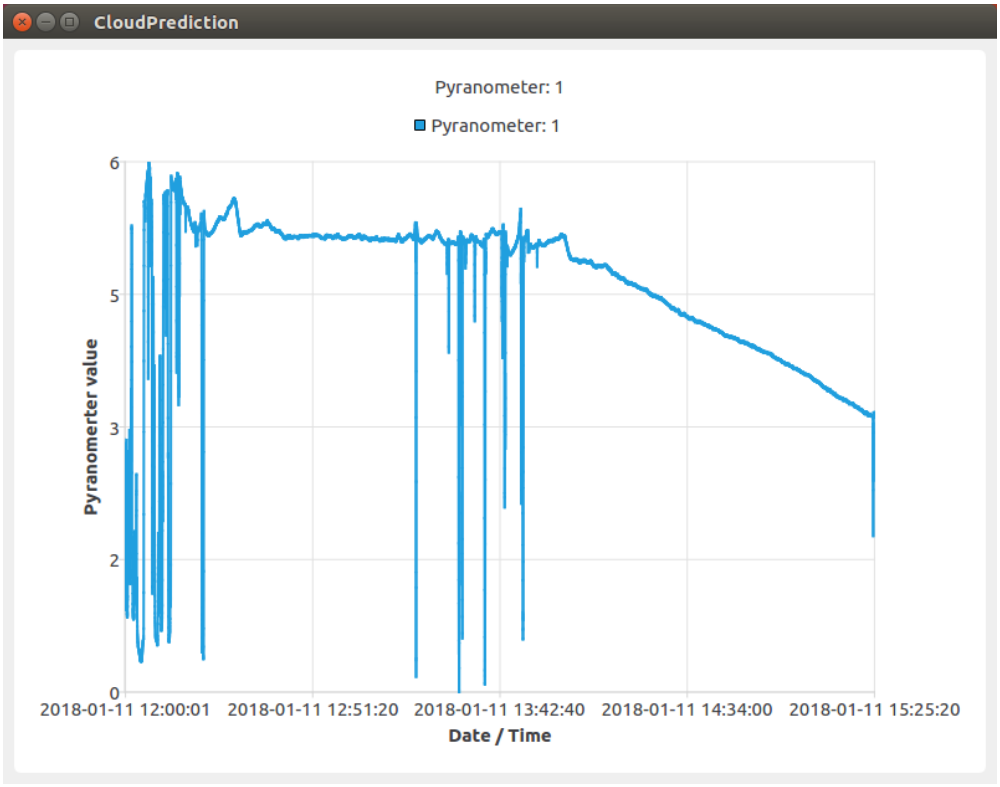


Figure 3.13: Presented data from pyranometer number 2 from 11-January-2018.

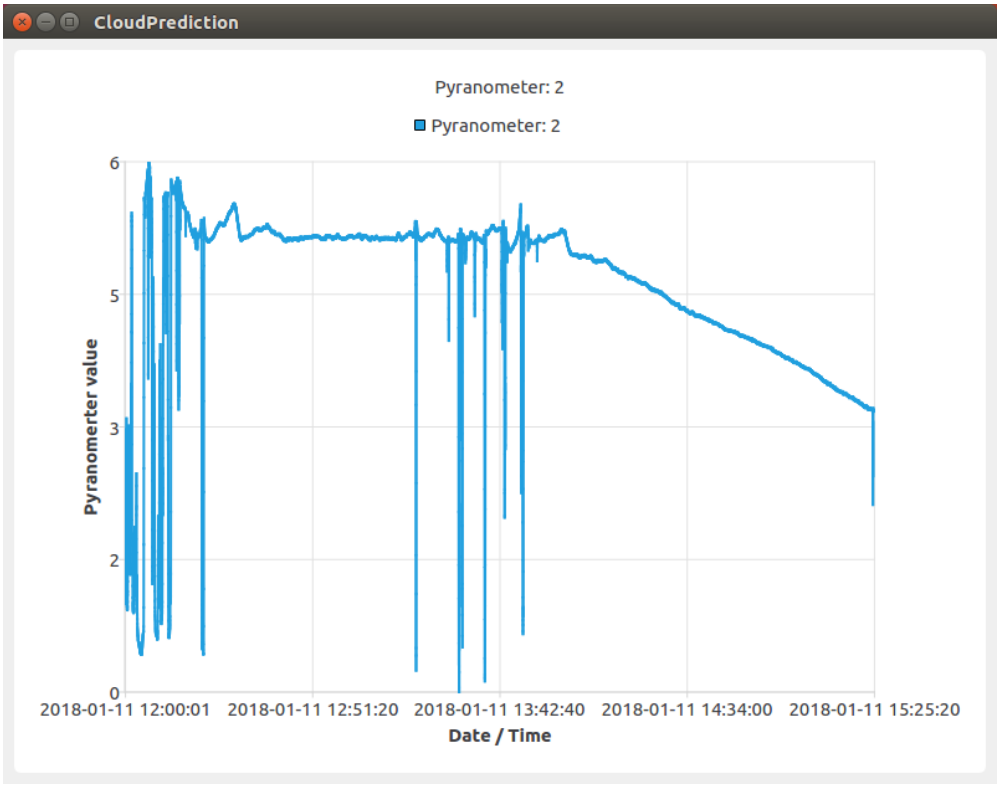


Figure 3.14: Presented data from pyranometer number 3 from 11-January-2018.

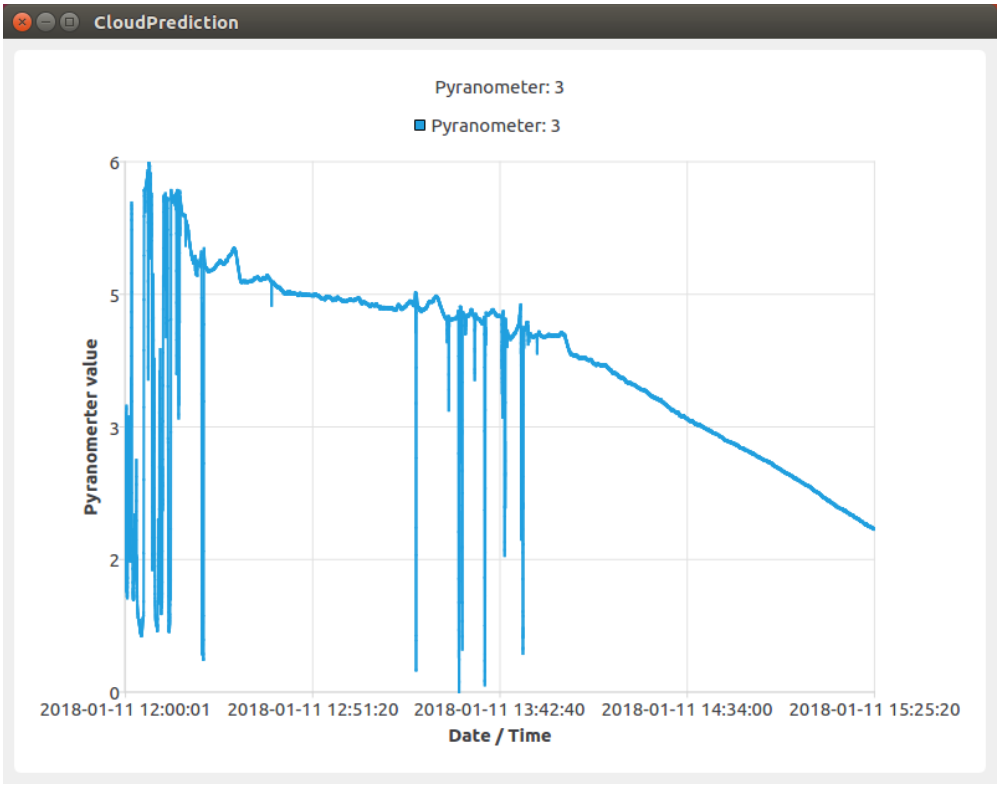


Figure 3.15: Presented data from pyranometer number 4 from 11-January-2018.

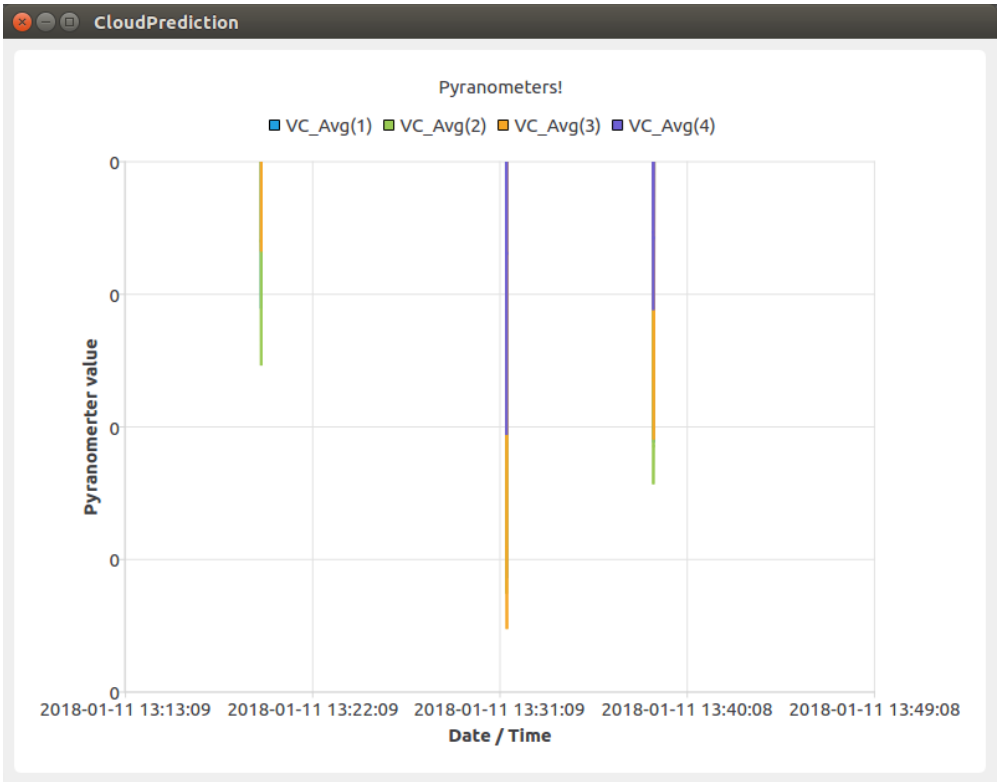


Figure 3.16: Example of zoom into data.

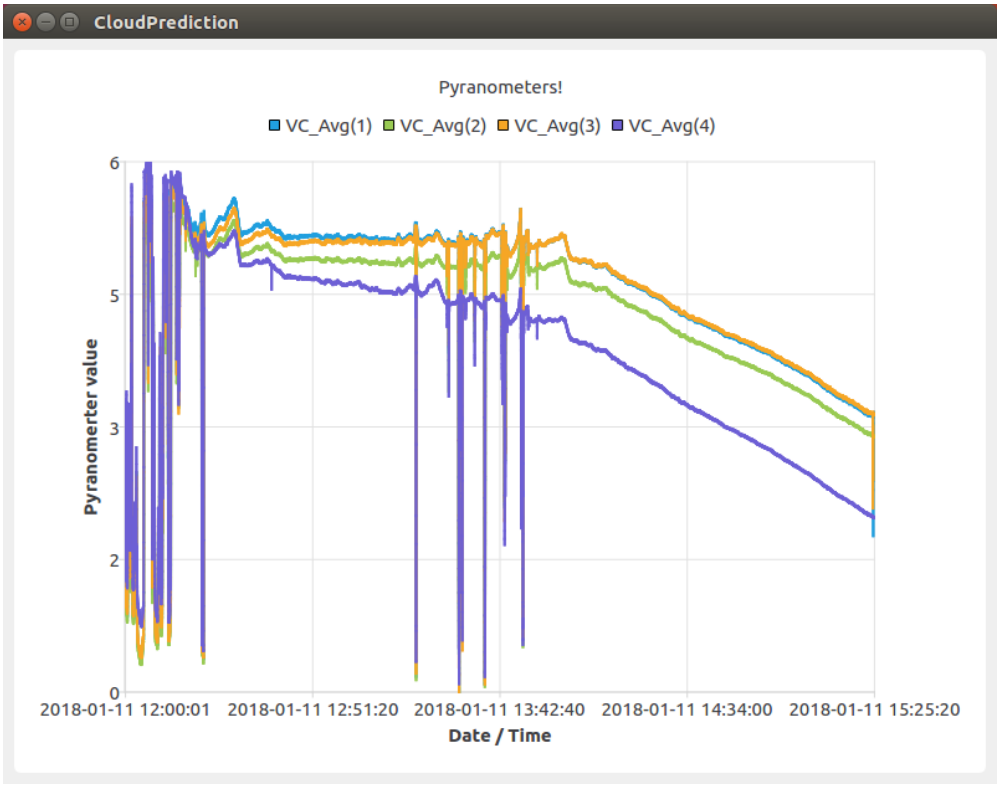


Figure 3.17: Presented measured data from all the pyranometers from 11-January-2018.

Chapter 4

Results

We have gathered full data (as images and voltage measurements from pyranometers) from two days: January 11th and 12th of 2018. Both "bigger" charts from these days are presented in figures: 4.1 and 4.2 (on pages 39 and 40 respectively).

Unfortunately (as stated in chapter 2) the only valid images are from January 11th. This fact is visible on the chart from January 12th (figure: 4.2 on page 40) where there is no a visible drop in voltage measurements (like in the chart from the previous day). Because of that, it is very hard to detect one solid "cloud". Moreover, on the first day we have wrongly set up the frequency of taking pictures: we took photos every second - 1Hz. And because of that error, it is almost impossible to detect the same cloud on two consecutive images.

Figure 4.3 (on page 41) shows a picture of our working envoironment of pyranometers to gather data. It is worth noticing that the amount of available pyranometers (only 4, set up as "North", "Central", "South" and "West" [placed on the wall]) did not allow to use the method described in subsection 3.1.1 (method defined on page: 20). The pyranometers are connected to a data logger configured to take a sample every second. Those samples are stored in the data logger and accessible by FTP through a local network. The working environment includes the Raspberry Pi and the camera described in Chapter 2. The Raspberry Pi periodically switches from *eduroam* to the local network to either upload pictures or read the data logger to later send the values read.

After gathering data from the pyranometers, local minimal values are computed (filtered

by using class persistence1d[19]). Listing 4.1 (on page 36) shows the data from the northern pyranometer.

Persistence : "sensitivity" value for searching for a local minimum in a selected set of data.

minimum index : index in table of data for local minimum,

TIME : shows exact date and time of that minimum value.

```
Persistence: 2.147 minimum index: 5522 TIME: czw. sty 11 13:32:03 2018
Persistence: 2.24 minimum index: 842 TIME: czw. sty 11 12:14:03 2018
Persistence: 2.333 minimum index: 570 TIME: czw. sty 11 12:09:31 2018
Persistence: 2.365 minimum index: 374 TIME: czw. sty 11 12:06:15 2018
Persistence: 2.602 minimum index: 875 TIME: czw. sty 11 12:14:36 2018
Persistence: 3.036 minimum index: 6511 TIME: czw. sty 11 13:48:32 2018
Persistence: 3.136 minimum index: 590 TIME: czw. sty 11 12:09:51 2018
Persistence: 3.223 minimum index: 6236 TIME: czw. sty 11 13:43:57 2018
Persistence: 3.484 minimum index: 12298 TIME: czw. sty 11 15:24:59 2018
Persistence: 3.606 minimum index: 444 TIME: czw. sty 11 12:07:25 2018
Persistence: 4.245 minimum index: 1259 TIME: czw. sty 11 12:21:00 2018
Persistence: 4.465 minimum index: 31 TIME: czw. sty 11 12:00:32 2018
Persistence: 4.603 minimum index: 5540 TIME: czw. sty 11 13:32:21 2018
Persistence: 4.906 minimum index: 6537 TIME: czw. sty 11 13:48:58 2018
Persistence: 5.142 minimum index: 716 TIME: czw. sty 11 12:11:57 2018
Persistence: 5.16 minimum index: 5911 TIME: czw. sty 11 13:38:32 2018
Persistence: 5.172 minimum index: 4780 TIME: czw. sty 11 13:19:41 2018
Persistence: 5.25 minimum index: 1282 TIME: czw. sty 11 12:21:23 2018
Persistence: 5.38 minimum index: 524 TIME: czw. sty 11 12:08:45 2018
Persistence: 5.697 minimum index: 257 TIME: czw. sty 11 12:04:18 2018
```

Listing 4.1: Local minima from "North" pyranometer.

The implemented method (described in subsection 3.1.2 on page 20) uses two pairs of sensors and each of those pairs share one common device In article[7] the authors used sensors on the "North", "Central" and "East" side. We had to change the configuration of the pyranometers due to space constrains. Thus, we replaced the "East" device with the "West" one, as shown in figure: 4.4 (on page 41) and picture 4.3 (on page 41).

Based on the proposed method, we selected the filtered values from the pyranometers which lay on a common narrow window (the time difference between values is not greater

than 7s). After that, we filtered pairs which happened in exactly the same time, because that would mean that the cloud has an undefined angle and infinitive speed. The result is shown in the listing 4.2 (on page 37).

```
#Unfiltered values
NC: ((31,31), (257,257), (374,374), (444,444), (524,524), (569,570),
    (590,590), (716,716), (842,842), (875,875), (1259,1259), (1282,1282)
    , (4780,4780), (5540,5540), (5910,5911), (6236,6236), (6511,6511),
    (6537,6537))
WC: ((31,31), (262,257), (374,374), (443,444), (523,524), (569,570),
    (589,590), (717,716), (842,842), (875,875), (1258,1259), (1281,1282)
    , (4779,4780), (5540,5540), (5910,5911), (6236,6236), (6511,6511),
    (6537,6537))
#Filtered values\label{lst:PairData2}
NC: ((257,257), (444,444), (524,524), (569,570), (590,590), (716,716),
    (1259,1259), (1282,1282), (4780,4780), (5910,5911), (6511,6511))
WC: ((262,257), (443,444), (523,524), (569,570), (589,590), (717,716),
    (1258,1259), (1281,1282), (4779,4780), (5910,5911), (6511,6511))
```

Listing 4.2: Indexes pairs of minimal values unfiltered and filtered.

Unfortunately, very likely due to our sampling resolution (1Hz), the local minimum of the different nodes happened exactly at the same time, so our results are mostly undefined as shown in table 4.1 (on page 37). The only valid values are in row 3 and 9 - for **Angle:** 1.5708 **Speed:** -5 and for **Angle:** 1.5708 **Speed:** -5. All other values have undefined speed and/or angle. The acceptable outcomes are presented in figure 4.5 (on page 42).

Table 4.1: Results of calculations the angle and speed of clouds.

Angle	**Value**	**Speed**	**Value**
Angle:	1.3734	Speed:	inf
Angle:	-0.785398	Speed:	-inf
Angle:	1.5708	Speed:	-5
Angle:	0	Speed:	nan
Angle:	0.785398	Speed:	inf
Angle:	0.785398	Speed:	inf
Angle:	-0.785398	Speed:	-inf
Angle:	-0.785398	Speed:	-inf
Angle:	1.5708	Speed:	-5
Angle:	nan	Speed:	nan
Angle:	nan	Speed:	nan

The "valid" values are from time indexes: (524,524), (523,524) and (4780,4780), (4799,4780) that indicates time of first one at: 12:08:43 and 12:08:44 and for the second one at: 13:19:39 and 13:19:40. The "lower values" of passing clouds can be visible on chart from that day (figure 4.1 on page 39) and approximations of that period of time on figures 4.6 and 4.7 (on page 42). What is more, the photos from that time (figure: 4.8 and 4.9 on page 43) indicates that in that period big clouds were moving above the sensors. Sadly on the "around" (in time) photos there are other clouds - there cannot prove the direction of clouds.

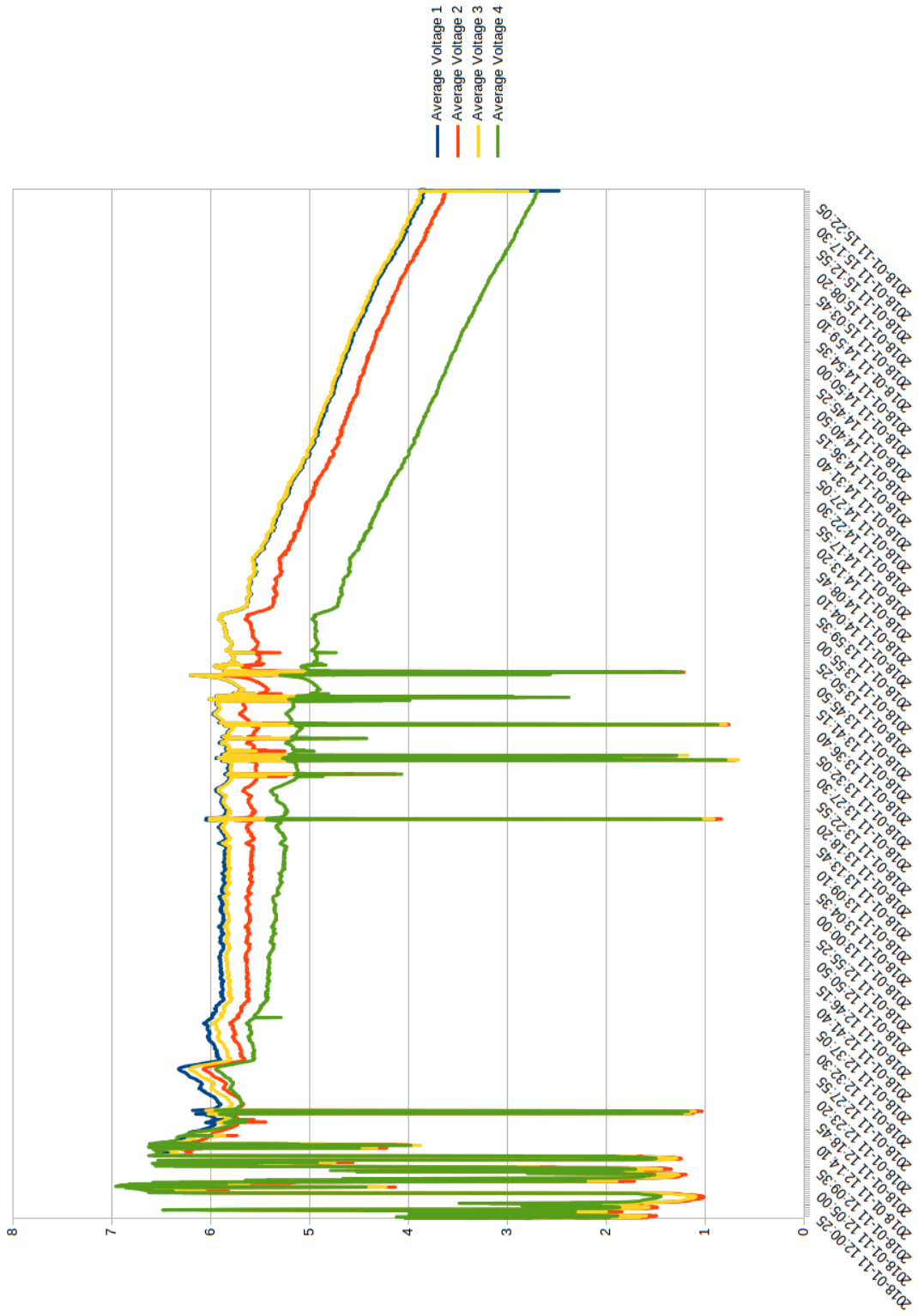


Figure 4.1: Combined chart of pyranometers output from 11-January-2018.

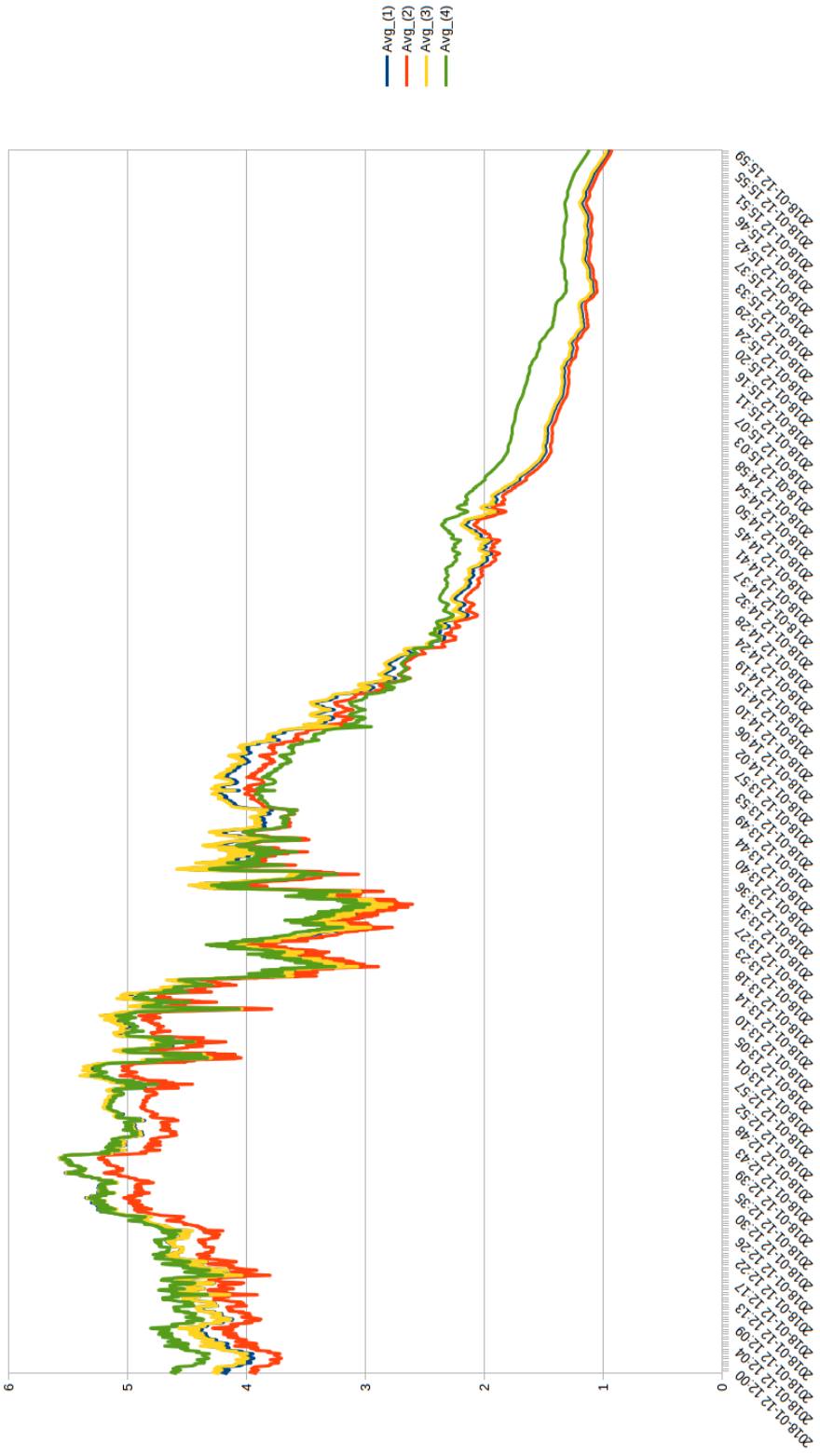


Figure 4.2: Combined chart of pyranometers output from 12-January-2018.

Figure 4.3: Photo of the set up Pyranometers.

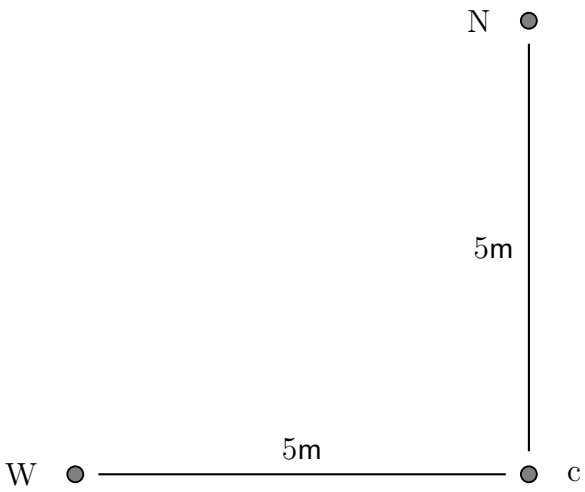


Figure 4.4: Draft of the position set of pyranometers.

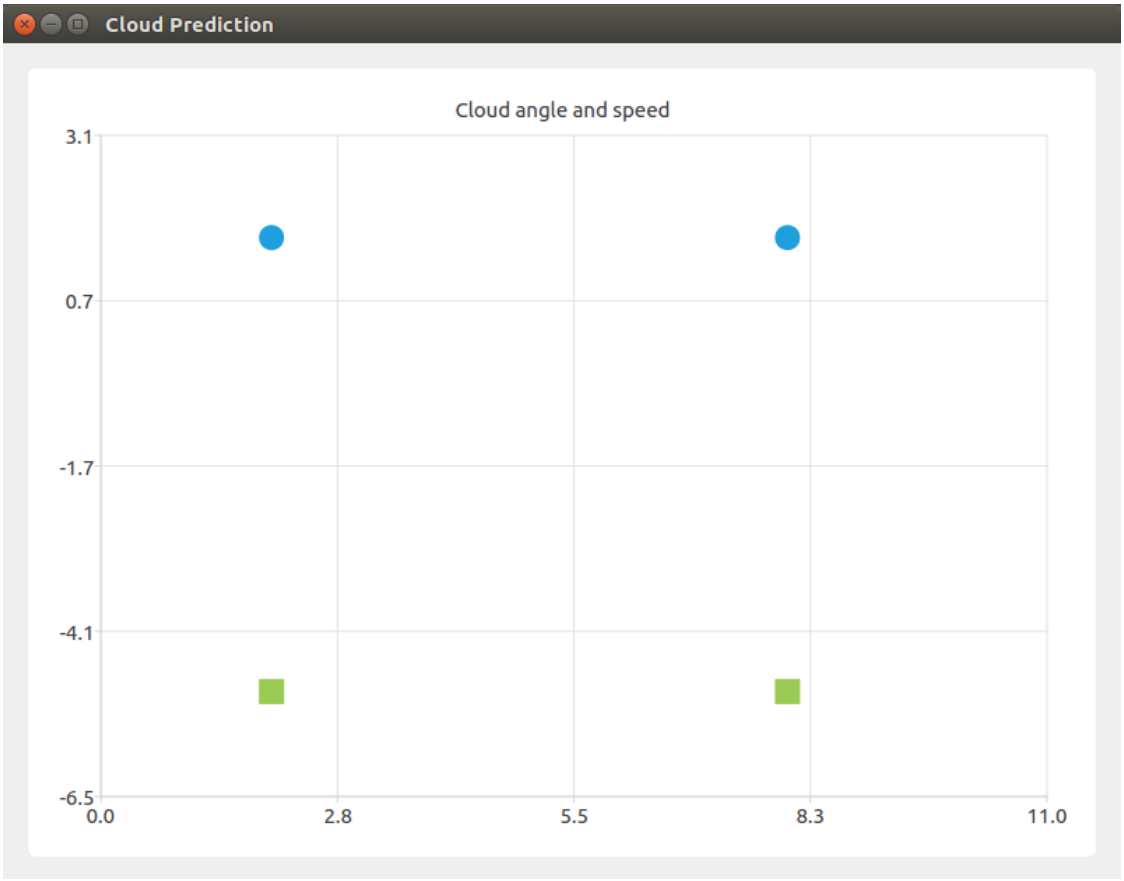


Figure 4.5: Print screen of window with "results" of calculation. The green squares indicates speed ($-5^{m}/_{s}$) of the clouds and blue circles indicates the angles of the clouds ($\approx 1.57°$ from the axis of the "North-Central" line).

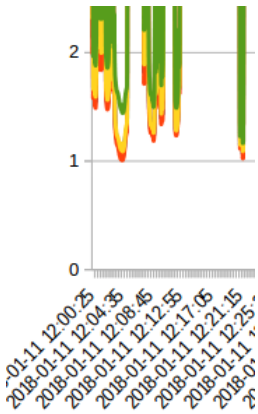


Figure 4.6: Fragment of measurement chart from 11 January with values "around" 12:08.

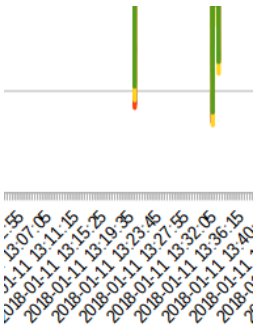


Figure 4.7: Fragment of measurement chart from 11 January with values "around" 13:19.

Figure 4.8: Photo taken at 12:09:14.

Figure 4.9: Photo taken at 13:20:13.

Chapter 5

Conclusions and future work

In my opinion, both this project and the topic that it addresses have a great potential. Ability to detect and predict the movement of the clouds is very interesting. I think that if we had had more time and more resources we could have obtained better results, but as of this 'product', I am pleased. With the proper frequency of gathering data from pyranometers and taking pictures of about 4 per second or more - $\geq 4\mathsf{Hz}$ - we could more effectively and with more accuracy accomplish this task.

Another problem was with calibration of pyranometers, as it can be seen on charts in figures 4.1 and 4.2 (pages 39 and 40 respectively). In those figures, it can be seen that the values are not fully overlapping to each other with the same "irradiation". Furthermore, more readings and analysis would be required to guarantee that the layout of the pyranometers is not introducing noise in the measurements.

In the future, this project can be upgraded by improving the frequency of gartering data and adding "fisheye" lens to the camera. The additional improvement would be adding the functionality of "online" prediction based for example on the data from Mosquitto broker, for which I was given an access for the MQTT server to establish Mosquitto[4] broker. I also prepared the main program to access this future feature by designing the data and usage an Qt MQTT module.

In my opinion, during work on this project and thesis, I have gained many abilities: to choose, read and understand complicated researches and to create complex programs. This

project can be updated and upgraded, for example to make it multiplatform, or to add an ability to predict clouds movement in real time while storing data for further off-line analysis.

Bibliography

[1] Arduino. Arduino - home. https://www.arduino.cc/. [Online; accessed 25-January-2018].

[2] T. Tooman C. N. Long, D. W. Slater. Total sky imager model 880 status and testing results. November 2001.

[3] J. Chater. A history of nuclear power. 2005.

[4] Eclipse Foundation. Mosquitto an open source mqtt v3.1/v3.1.1 broker. https://mosquitto.org/.

[5] Raspberry Pi Foundation. Raspberry pi camera module. https://www.raspberrypi.org/documentation/hardware/camera/README.md. [Online; accessed 2-February-2018].

[6] Qasim Sattar Sadaqat-ur-Rehman Amjad Ali Irfanullah, Kamal Haider. An efficient approach for sky detection. *IJCSI International Journal of Computer Science*, 10:222–226, July 2013.

[7] J. Kleissl J.L. Bosch, Y. Zheng. Deriving cloud velocity from an array of solar radiation measurements. *Solar Energy*, 87:196–203, 2013.

[8] Y.Jun L. Qingyong, L. Weitao. A hybrid thresholding algorithm for cloud detection on ground-based color images. 2011.

[9] Environmental Measurements Limited. Solar radiation measurement - sks1110.

[10] Y. Nishiwaki. Fifty years after hiroshima and nagasaki.

[11] Office of Energy Efficiency and Renewable Energy U.S. Department of Energy. Renewable electricity generation.

[12] Harvard College Review of Environment and Society. The future of nuclear energy. Spring 2015.

[13] P. Parysek. A system for 3d reconstruction and medical analysis of the lower limbs. August 2016.

[14] User: rca. Cloud extraction from sky image. http://answers.opencv.org/question/91752/cloud-extraction-from-sky-image/.

[15] S. Winkler S. Dev, Y. Hui Lee.

[16] OpenCV team. Opencv. https://opencv.org/. [Online; accessed 2-February-2018].

[17] Sir P. Nurse the Royal Society, Dr. R. J. Cicerone. Climate change evidence and causes.

[18] thediylife http://www.the-diy-life.com. Arduino solar tracker (single or dual axis). http://www.instructables.com/id/Arduino-Solar-Tracker-Single-or-Dual-Axis/. [Online; accessed 25-January-2018].

[19] T. Weinkauf Y. Kozlov. Extracting and filtering minima and maxima of 1d functions. https://people.mpi-inf.mpg.de/~weinkauf/notes/persistence1d.html.

[20] G. Peng Y. Zheng, X. Zhang. Cloud motion derivation with a pyranometer array measurements. *International Journal of Computer and Electrical Engineering*, 5:183–186, 2013.